Report | HW 5

CS663 - Digital Image Processing

Omkar Shirpure (22B0910)
Suryansh Patidar (22B1036)
Aditya Saran (22B1839)

Contents

1 Q1 . . . **1**
1.1 a. . . . 1
1.2 b. . . . 1

2 Q2 . . . **2**

3 Q3 . . . **4**

4 Q4 . . . **6**
4.1 Part(a) . . . 6
4.2 Part (b): Minimum Value of m . . . 7
4.3 Part (c): When the Locations of Non-Zero Elements Are Unknown . . . 7

5 Q5 . . . **9**
5.1 Part(a): Why this solution fails? . . . 9
5.2 Part(b) . . . 9
5.3 Part(c): Minimizing $E(R)$ and Maximizing trace$(P_1^T R P_2)$ . . . 10
5.4 Part(d) . . . 11
5.5 Part(e) . . . 12
5.6 Part(f) . . . 12
5.7 Part(g) . . . 13

1 Q1

1.1 a.

The paper describes a Fourier domain approach for determining translation between two images. Here's a refined explanation of the process:

To register two images using this approach, the images are first transformed into the *Fourier domain* by applying the Fourier transform. This allows translation between the images to be determined through *phase matching* rather than direct comparison in the spatial domain.

1. *Translation Model*: Suppose two images $f_1(x, y)$ and $f_2(x, y)$ differ only by a translation. The relationship can be written as:

$$f_2(x, y) = f_1(x - x_0, y - y_0)$$

where (x_0, y_0) represents the translation offsets.

2. *Fourier Transform and Phase Shift*: Taking the Fourier transform of both images introduces an extra phase factor in f_2:

$$F_2(u, v) = e^{-j2\pi(ux_0+vy_0)} \cdot F_1(u, v)$$

where $F_1(u, v)$ and $F_2(u, v)$ are the Fourier transforms of f_1 and f_2, and (u, v) are the frequency domain variables.

3. *Cross-Power Spectrum: To extract the translation, we compute the **cross-power spectrum* of the two Fourier transforms:

$$\frac{F_1(u, v)F_2^*(u, v)}{|F_1(u, v)F_2(u, v)|} = e^{j2\pi(ux_0+vy_0)}$$

4. *Time Complexity*: Since this method involves calculating the 2D Fourier transform, its time complexity is $O(N^2 \log(N))$ for an image of size $N \times N$. This is significantly more efficient than a pixel-wise comparison for determining translation, which has a time complexity of $O(N^3)$. Therefore, using the Fourier domain approach provides a considerable computational advantage.

1.2 b.

To correct for rotation between two images, the paper presents a solution using the Fourier domain. Suppose f_2 is a rotated and translated version of f_1, with a rotation angle θ_0 and a translation offset (x_0, y_0). This relationship can be expressed as:

$$f_2(x, y) = f_1(x \cos(\theta_0) + y \sin(\theta_0) - x_0,\ y \cos(\theta_0) - x \sin(\theta_0) - y_0)$$

After applying the Fourier transform to both sides, we obtain:

$$F_2(u, v) = e^{-j2\pi(ux_0+vy_0)}F_1(u \cos(\theta_0) + v \sin(\theta_0),\ v \cos(\theta_0) - u \sin(\theta_0))$$

Focusing on the magnitudes of the Fourier transforms (which are rotation-invariant), we have:

$$M_2(u, v) = M_1(u \cos(\theta_0) + v \sin(\theta_0),\ v \cos(\theta_0) - u \sin(\theta_0))$$

Converting to polar coordinates (r, θ), this becomes a shift in the angular coordinate:

$$M_1(r, \theta) = M_2(r, \theta - \theta_0)$$

Thus, by analyzing the shift in the angular domain, the rotation θ_0 between the images can be identified and corrected.

2 Q2

To solve for f_1 and f_2 from the given images g_1 and g_2, we can follow the steps below.

Step 1: Given Equations

From the problem, we have:

$$g_1 = f_1 + h_2 * f_2,$$
$$g_2 = h_1 * f_1 + f_2,$$

where:

- g_1 and g_2 are the two images captured.
- f_1 represents the scene outside (in focus in g_1 but blurred in g_2).
- f_2 represents the reflection of the scene inside the room (blurred in g_1 and in focus in g_2).
- h_1 and h_2 are known blur kernels.

Step 2: Using Fourier Transform for Separation

To separate f_1 and f_2, a standard approach is to use the Fourier Transform, which converts convolutions into simple multiplications in the frequency domain. Let's denote the Fourier Transform of a function f as $\mathcal{F}(f)$.

Taking the Fourier Transform of both equations:

$$\mathcal{F}(g_1) = \mathcal{F}(f_1) + \mathcal{F}(h_2) \cdot \mathcal{F}(f_2),$$
$$\mathcal{F}(g_2) = \mathcal{F}(h_1) \cdot \mathcal{F}(f_1) + \mathcal{F}(f_2).$$

Let:

$$G_1 = \mathcal{F}(g_1),$$
$$G_2 = \mathcal{F}(g_2),$$
$$H_1 = \mathcal{F}(h_1),$$
$$H_2 = \mathcal{F}(h_2),$$
$$F_1 = \mathcal{F}(f_1),$$
$$F_2 = \mathcal{F}(f_2).$$

Then the equations become:

$$G_1 = F_1 + H_2 \cdot F_2,$$
$$G_2 = H_1 \cdot F_1 + F_2.$$

Step 3: Solve for F_1 and F_2

From the first equation, we can express F_1 as:

$$F_1 = G_1 - H_2 \cdot F_2.$$

Substituting F_1 from this expression into the second equation:

$$G_2 = H_1 \cdot (G_1 - H_2 \cdot F_2) + F_2.$$

Expanding and rearranging terms to isolate F_2:

$$G_2 = H_1 \cdot G_1 - H_1 \cdot H_2 \cdot F_2 + F_2,$$

$$G_2 - H_1 \cdot G_1 = (1 - H_1 \cdot H_2) \cdot F_2,$$

$$F_2 = \frac{G_2 - H_1 \cdot G_1}{1 - H_1 \cdot H_2}.$$

Then, we can substitute F_2 back to find F_1:

$$F_1 = G_1 - H_2 \cdot F_2.$$

To express f_1 and f_2 in terms of integrals, we start by taking the inverse Fourier transforms of F_1 and F_2 as follows:

Given:

$$F_2 = \frac{G_2 - H_1 \cdot G_1}{1 - H_1 \cdot H_2}$$

$$F_1 = G_1 - H_2 \cdot F_2$$

Expression for f_2 The inverse Fourier transform of F_2 is:

$$f_2(x, y) = \int\int \frac{G_2(u, v) - H_1(u, v) \cdot G_1(u, v)}{1 - H_1(u, v) \cdot H_2(u, v)} e^{j2\pi(ux+vy)} \, du \, dv$$

Expression for f_1 Using $F_1 = G_1 - H_2 \cdot F_2$, we obtain:

$$f_1(x, y) = \int\int \left(G_1(u, v) - H_2(u, v) \cdot \frac{G_2(u, v) - H_1(u, v) \cdot G_1(u, v)}{1 - H_1(u, v) \cdot H_2(u, v)} \right) e^{j2\pi(ux+vy)} \, du \, dv$$

In these expressions: - (u, v) are the frequency variables. - $G_1(u, v)$ and $G_2(u, v)$ are the Fourier transforms of the observed images g_1 and g_2. - $H_1(u, v)$ and $H_2(u, v)$ are the Fourier transforms of the blur kernels h_1 and h_2.

This formulation provides f_1 and f_2 in the spatial domain, given the frequency-domain representations of the images and blur kernels.

Step 4: Problematic Aspect of the Solution

The main issue with this solution is that it becomes **unstable when** $H_1 \cdot H_2 \approx 1$. In such cases, the denominator $1 - H_1 \cdot H_2$ approaches zero, leading to very large or undefined values for F_2 (and consequently for F_1).

This instability is due to the fact that if H_1 and H_2 are such that their product is close to unity, it becomes challenging to separate f_1 and f_2 distinctly in the frequency domain, as the captured images g_1 and g_2 do not provide sufficient independent information about each component.

3 Q3

Based on the provided paper, here is a detailed summary and analysis of its contributions to underwater image restoration.

Title, Venue, and Publication Year

- **Title**: Genetic Algorithm-based Dark Channel Prior Parameters Selection for Single Underwater Image Dehazing
- **Venue**: 2020 IEEE Region 10 Conference (TENCON)
- **Publication Year**: 2020

Problem Statement

The paper addresses the problem of *underwater image dehazing,* a significant challenge in underwater image processing. Underwater images typically suffer from low contrast, poor visibility, and color distortion due to light absorption and scattering in water. To enhance these images, the paper utilizes the **Dark Channel Prior (DCP)** algorithm, originally developed for land-based images, and optimizes it specifically for underwater conditions. However, the default DCP parameters may not be ideal for the unique properties of underwater scenes like illuminations and turbidity. Therefore, the paper proposes a **Genetic Algorithm (GA)** approach to automatically optimize the DCP parameters for each image, leading to better restoration results tailored to underwater conditions.

Dark Channel Prior (DCP) for Image Dehazing

The Dark Channel Prior (DCP) is an image dehazing algorithm originally developed for enhancing hazy outdoor images. The core idea behind DCP is based on the observation that in most non-sky patches of outdoor images, there exists at least one color channel (Red, Green, or Blue) with very low intensity, leading to dark pixels in these areas. This property, known as the "dark channel," can be used to estimate the haze in the image and recover the underlying scene.

Cost Function for Optimization

The cost function used to guide the optimization in this paper is based on **information entropy**, which measures the quality of the restored image. Information entropy quantifies the amount of information or detail present in the image, with higher entropy values indicating improved visibility and contrast.

1. **Information Entropy (H)** for each color channel (Red, Green, and Blue) is defined as:

$$H(x)c \in \{R, G, B\} = -\sum i = 0^{255} p(x) \log_2 p(x)$$

where:

- $H(x)$ represents the entropy of the image.
- c denotes each color channel (Red, Green, and Blue).
- $p(x)$ is the probability distribution of pixel intensity levels within each channel.

2. **Root Mean Square (RMS) Entropy for the Color Image** combines the entropy of each color channel to provide a single measure of information content across the color image:

$$\bar{H}(x) = \sqrt{\frac{H(x)_R^2 + H(x)_G^2 + H(x)_B^2}{3}}$$

where:

- $H(x)_R$, $H(x)_G$, and $H(x)_B$ are the entropies for the Red, Green, and Blue channels, respectively.
- The RMS entropy value, $\bar{H}(x)$, indicates the overall information content across all color channels, with higher values indicating clearer, more detailed restored images.

Optimization Parameters

The Genetic Algorithm optimizes two critical DCP parameters:

- ω: Controls the strength of the transmission map, with permissible values in the range $0 \leq \omega \leq 1$.
- t_0: Sets the lower bound of the transmission, within the range $0.1 \leq t_0 < 0.95$.

If the GA explores parameter values outside these ranges, a penalty is applied to the fitness score, ensuring the search stays within practical limits for enhancing underwater images.

Genetic Algorithm (GA) Methodology

The Genetic Algorithm proceeds by iteratively improving a population of solutions (parameter sets) using selection, crossover, and mutation. The fitness function, based on RMS entropy, guides the search for optimal parameters that maximize image quality by improving contrast, color accuracy, and clarity in underwater images.

Summary

In conclusion, this paper introduces a GA-based approach for optimizing the DCP algorithm, enhancing the visibility and quality of underwater images. By maximizing information entropy, this method adaptively tunes the DCP parameters to effectively restore underwater images with better color balance, contrast, and detail. The optimized DCP parameters significantly outperform the default settings, making this approach suitable for various underwater scenes.

4 Q4

4.1 Part(a)

4.1.1 Given:

An $n \times n$ image $f(x, y)$ with only k non-zero elements, where $k \ll n^2$. The locations of the k non-zero elements are known, as well as the locations of the $n^2 - k$ zero elements. We have only m DFT coefficients (with $m < n^2$), which we aim to use to reconstruct the image.

4.1.2 Approach Using Known Zero Locations and DFT Coefficients

The DFT of an image $f(x, y)$ is given by:

$$F(u, v) = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} f(x, y) e^{-j2\pi\left(\frac{ux}{n} + \frac{vy}{n}\right)}$$

where $F(u, v)$ are the DFT coefficients of $f(x, y)$.

Since the DFT is linear, we can express this transformation in matrix form. Let f be a vectorized form of the image (flattened into a column vector of length n^2), and let F be the vector of DFT coefficients. The DFT operation can be represented as:

$$F = Af$$

where A is the $n^2 \times n^2$ DFT matrix that transforms f into its Fourier coefficients F.

4.1.3 Step 1: Leveraging Known Zero Locations

Since we know the locations of the zero elements in $f(x, y)$, we can express these zero constraints as a system of linear equations. Let's define a selection matrix Z that picks out the indices corresponding to the known zero entries in f. This gives us:

$$Zf = 0$$

where Z is a binary matrix of size $(n^2 - k) \times n^2$ that has rows corresponding to the zero entries of f.

4.1.4 Step 2: Setting Up an Overdetermined System Using Known DFT Coefficients

We only know m of the DFT coefficients, where $m < n^2$. Let A_m represent the $m \times n^2$ matrix consisting of the rows of A that correspond to the known DFT frequencies. Then, we can write:

$$F_m = A_m f$$

where F_m is the vector of the m known DFT coefficients.

4.1.5 Step 3: Combining Constraints and Solving Using the Pseudo-Inverse

We now have two sets of equations:

- The zero constraints: $Zf = 0$
- The known DFT coefficients: $A_m f = F_m$

We can combine these into a single system:

$$[A_m \; Z]^T f = [F_m \; 0]^T$$

Let $B = [A_m \; Z]^T$ and $g = [F_m \; 0]^T$. The system becomes:

$$Bf = g$$

Since $m + (n^2 - k) \geq n^2$, this system is generally overdetermined. We can solve it using the pseudo-inverse:

$$f = B^+ g$$

where B^+ is the Moore-Penrose pseudo-inverse of B. This solution minimizes the least-squares error, effectively using the known DFT coefficients and zero locations to reconstruct the image.

4.2 Part (b): Minimum Value of m

To uniquely determine f, the system needs to be at least determined, so we require:

$$m + (n^2 - k) \geq n^2$$

which simplifies to:

$$m \geq k$$

Thus, the minimum value of m required is k.

4.3 Part (c): When the Locations of Non-Zero Elements Are Unknown

Now, let's explore why the reconstruction approach fails if the locations of the k non-zero elements are unknown.

4.3.1 1. Loss of Zero Constraints

When the locations of the non-zero elements are unknown:

We no longer know which specific pixels in $f(x, y)$ are zero. This removes the ability to use the zero constraints we used in parts (a) and (b).

In the previous approach, knowing that $n^2 - k$ pixels were zero allowed us to set up a system of linear equations with constraints on those zero pixels, which helped reduce the number of unknowns. Without these zero constraints, the problem becomes much less structured and challenging to solve with limited DFT data.

4.3.2 2. Increased Solution Ambiguity

Without knowledge of the zero locations, our only available information comes from the m DFT coefficients:

$$A_m f = F_m$$

where A_m is an $m \times n^2$ matrix. Since $m < n^2$, this system is underdetermined in general because there are n^2 unknowns in f and only m equations.

The key difficulty is that without knowing the zero locations, we have to consider all n^2 pixels in f as potentially non-zero. Therefore:

- We now have n^2 potential unknowns instead of just k non-zero values at specific positions.
- The system becomes fundamentally underdetermined because $m < n^2$, and there are not enough equations to solve for all n^2 unknowns.

5 Q5

5.1 Part(a): Why this solution fails?

5.1.1 1. Orthonormality Constraint

The matrix R is required to be orthonormal, meaning it must satisfy:

$$R^T R = I$$

However, the least squares solution $R = P_1 P_2^T (P_2 P_2^T)^{-1}$ does not guarantee that R will be orthonormal. This is because the least squares solution is focused on minimizing the error $\|P_1 - RP_2\|_F$, but it does not enforce the orthonormality constraint on R.

Thus, even though the least squares method minimizes the error, the resulting matrix R may not satisfy $R^T R = I$, which is a crucial condition in this problem.

5.1.2 2. Effect of Noise E

The given relationship between P_1 and P_2 also includes a noise term E, i.e.,

$$P_1 = RP_2 + E$$

The presence of noise implies that the true relationship between P_1 and P_2 is not perfectly captured by the matrix R. The least squares method minimizes the error term $\|E\|_F^2$, but it does not account for the fact that P_1 and P_2 are related through a noisy transformation.

As a result, the least squares solution might not capture the correct orthonormal transformation due to the noise, and the resulting R may not be the true orthonormal transformation matrix.

5.1.3 3. Rank Deficiency

If P_2 is not of full rank, the matrix $P_2 P_2^T$ will be singular, and its inverse $(P_2 P_2^T)^{-1}$ will not exist. In such cases, the least squares solution is ill-defined because matrix inversion is not possible.+ Thus, if P_2 is rank-deficient, the least squares approach will fail to provide a valid solution for R.

Therefore, a more suitable method, such as Singular Value Decomposition (SVD) or Procrustes analysis, should be used to solve this problem while ensuring the orthonormality of R.

5.2 Part(b)

We seek to minimize the following objective function:

$$E(R) = \|P_1 - RP_2\|_F^2$$

Expanding the squared Frobenius norm, we get:

$$E(R) = \text{trace}\left((P_1 - RP_2)^T (P_1 - RP_2)\right)$$

$$= \text{trace}(P_1^T P_1 - P_1^T R P_2 - P_2^T R^T P_1 + P_2^T R^T R P_2)$$

Simplifying this to get Eq(4) by using the fact that R is orthonormal, which implies $R^T R = I$

$$= \text{trace}(P_1^T P_1 - P_1^T R P_2 - P_2^T R^T P_1 + P_2^T P_2)$$

$$= \text{trace}(P_1^T P_1) - \text{trace}(P_1^T R P_2) - \text{trace}(P_2^T R^T P_1) + \text{trace}(P_2^T P_2)$$

Now to get Eq(5), we know that trace(A) = trace(A^T) because the diagonal values do not change if we take the transpose. And we see that

$$(P_2^T R^T P_1)^T = P_1^T R P_2$$

So,

$$\text{trace}(P_2^T R^T P_1) = \text{trace}(P_1^T R P_2)$$

Thus, the expression becomes:

$$E(R) = \text{trace}(P_1^T P_1) + \text{trace}(P_2^T P_2) - 2 \cdot \text{trace}(P_1^T R P_2)$$

$$E(R) = \text{trace}(P_1^T P_1 + P_2^T P_2) - 2 \cdot \text{trace}(P_1^T R P_2)$$

5.3 Part(c): Minimizing $E(R)$ and Maximizing trace$(P_1^T R P_2)$

We seek to minimize the objective function:

$$E(R) = \text{trace}(P_1^T P_1) + \text{trace}(P_2^T P_2) - 2 \cdot \text{trace}(P_1^T R P_2)$$

The first two terms, trace$(P_1^T P_1)$ and trace$(P_2^T P_2)$, are constants with respect to R. Therefore, minimizing $E(R)$ is equivalent to minimizing the term:

$$-2 \cdot \text{trace}(P_1^T R P_2)$$

Thus, the minimization of $E(R)$ reduces to the maximization of the term trace$(P_1^T R P_2)$.

5.4 Part(d)

To prove trace$(P_1^T R P_2)$ = trace$(R P_2 P_1^T)$, we have to prove the cyclic property of trace. I.e.

$$\text{trace}(ABC) = \text{trace}(BCA) = \text{trace}(CAB)$$

Proof of the Cyclic Property of the Trace
The trace of the matrix product ABC can be written as:

$$\text{trace}(ABC) = \sum_i (ABC)_{ii}$$

where $(ABC)_{ii}$ denotes the i-th diagonal element of the matrix product ABC.
The element at position (i, j) of the matrix product ABC is given by:

$$(ABC)_{ij} = \sum_k A_{ik} B_{kj} C_{jl}$$

For the diagonal elements, we set $i = j$, so the diagonal element of ABC is:

$$(ABC)_{ii} = \sum_k A_{ik} B_{kl} C_{li}$$

Thus, the trace of ABC becomes:

$$\text{trace}(ABC) = \sum_i \sum_k A_{ik} B_{kl} C_{li}$$

Now, we apply cyclic permutations of the matrices inside the sum. The key observation is that the trace is invariant under cyclic permutations of the matrices in the product. Let us now show this explicitly for the traces of CAB and BCA.
Similarly the trace of CAB is:

$$\text{trace}(CAB) = \sum_i \sum_k C_{ik} A_{kl} B_{li}$$

and the trace of BCA is:

$$\text{trace}(BCA) = \sum_i \sum_k B_{ik} C_{kl} A_{li}$$

we observe that all three expressions for the trace are of the same form, with different orderings of the matrices:

$$\text{trace}(ABC) = \sum_i \sum_k A_{ik} B_{kl} C_{li}$$

$$\text{trace}(CAB) = \sum_i \sum_k C_{ik} A_{kl} B_{li}$$

$$\text{trace}(BCA) = \sum_i \sum_k B_{ik} C_{kl} A_{li}$$

Since matrix multiplication is associative, and the summation is over the same set of indices i and k, we conclude that all three traces are identical:

$$\text{trace}(ABC) = \text{trace}(CAB) = \text{trace}(BCA)$$

So the matrix $R P_2 P_1^T$ is the cyclic form of matrix $P_1^T R P_2$. Hence according to the cyclic property of trace

$$\text{trace}(P_1^T R P_2) = \text{trace}(R P_2 P_1^T)$$

5.5 Part(e)

We want to maximize the following quantity:

$$\text{trace}(S'X)$$

where S' is a diagonal matrix, and $X = V'^T RU'$.

Step 1: **Objective Function** We are given:

$$X = V'^T RU'$$

and the goal is to maximize trace$(S'X)$. Therefore, the objective is to maximize:

$$\text{trace}(S'V'^T RU')$$

Since S' is diagonal, we can express it as $S' = \text{diag}(s_1, s_2, \ldots, s_n)$, where s_i are the diagonal elements of S'. Thus, we can write the trace expression as:

$$\text{trace}(S'V'^T RU') = \sum_{i=1}^{n} s_i \left(V'^T RU'\right)_{ii}$$

Step 2: **Cyclic Property of the Trace** Using the cyclic property of the trace, we can rewrite the expression as:

$$\text{trace}(S'V'^T RU') = \text{trace}(V'^T RU'S')$$

Now, the objective is to maximize trace$(V'^T RU'S')$. Since S' is diagonal, we are essentially maximizing the sum of the diagonal elements of the matrix product $V'^T RU'S'$.

Step 3: **Maximizing the Trace** Now, observe that:

- V' and U' are orthonormal matrices (from the SVD of $P_2 P_1^T$).
- R is an orthonormal matrix.

To maximize the trace, we need to choose R such that the diagonal entries of $RU'S'V'^T$ are maximized. This is achieved when the columns of U' and V' are aligned, which leads to:

$$R = V'U'^T$$

Thus, we have:

$$X = V'^T (V'U'^T)U' = V'^T V'U'^T U' = I$$

Step 4: **Conclusion** The matrix X that maximizes trace$(S'X)$ is the identity matrix I. Therefore, the value of X that maximizes the trace is:

$$X = I$$

5.6 Part(f)

Step 1: **Recall the relationship between X and R**

From the previous derivation, we have the equation:

$$X = V'^T RU'$$

Since we determined that $X = I$ (the identity matrix), we can substitute this into the equation:

$$I = V'^T RU'$$

Step 2: **Solve for R**

To solve for R, we multiply both sides of the equation by $(V'^T)^{-1}$ on the left and U'^{-1} on the right. Since V' and U' are orthonormal matrices (i.e., $V'^T V' = I$ and $U'^T U' = I$), we can simplify the expression:

$$V'^T R U' = I$$

Multiplying by V' on the left and U'^T on the right:

$$R = V' U'^T$$

5.7 Part(g)

To impose the constraint that R is specifically a rotation matrix, you need to ensure that R satisfies the following properties:

1. Orthonormality: The matrix R must be orthonormal. This means that R must satisfy the condition:

$$R^T R = I$$

which ensures that the columns of R form an orthonormal set, i.e., they are both orthogonal and of unit length.

2. Determinant Condition: For R to be a rotation matrix, in addition to being orthonormal, it must have a determinant of $+1$. This ensures that the matrix represents a proper rotation (not a reflection). The condition is:

$$\text{det}(R) = 1$$

This condition excludes improper orthogonal matrices, which represent reflections, and ensures that the transformation represented by R preserves orientation and scales objects uniformly.

To summarize, if you want R to be a rotation matrix, you need to impose the following two constraints:

$$R^T R = I \quad \text{(Orthonormality)}$$

$$\text{det}(R) = 1 \quad \text{(Proper rotation)}$$

These two constraints ensure that R is not only an orthonormal matrix but also represents a rotation (rather than a reflection or other transformation).